

Course on Analysis of Embedded Systems

Frits Vaandrager

Institute for Computing and Information Sciences
Radboud University Nijmegen

<http://www.cs.ru.nl/F.Vaandrager/>

Spring 2011

Reliability in System Design

- Computer systems are getting more complex and pervasive
- Safety-critical applications: bugs are unacceptable.
Mission control (ARIANE-5), medicine, automotive, etc
- Bugs are expensive: earlier we catch them, the better.
E.g. FDIV in Pentium
- Testing takes more time than designing. Automation key to improve time-to-market
- Increasing use of programmable components shifts focus from low-level optimizations to high-level designs

Goal Formal Verification

Provide tools and techniques as design aids to produce reliable systems

Coping with Complexity

- Design reuse
- Separation of concerns: logical vs physical, logical vs timing, etc
- Formalization – precise unambiguous semantics
- Abstraction – eliminate unnecessary details
- Decomposition – divide and conquer
- Incremental refinements

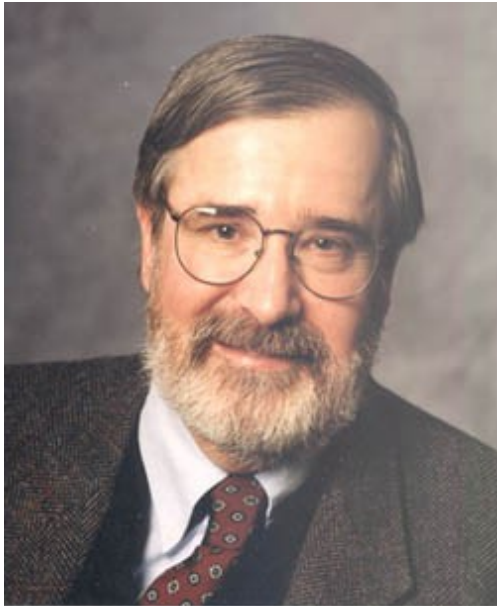
What is Formal Verification?

- Build mathematical *model* of system: what are *possible* behaviors?
- Write correctness requirements in a *specification* language: what are *desirable* behaviors?
- Analysis: check that model satisfies specification
- Formal \Rightarrow Correctness claim is precise mathematical statement
- Verification \Rightarrow Analysis either proves or disproves correctness claim

Alternative Approaches

- Testing: Execute the actual system on selected inputs
- Simulation: Simulate a model of the system on selected inputs (not exhaustive)

ACM Turing Award 2007 for Clarke, Emerson & Sifakis



”for their original and continuing research in a quality assurance process known as Model Checking. Their innovations transformed this approach from a theoretical technique to a highly effective verification technology that enables computer hardware and software engineers to find errors efficiently in complex system designs”

Algorithmic versus Interactive Verification

Algorithmic analysis

- Analysis performed by an algorithm (tool)
- Analysis gives counter-examples for debugging
- Typically requires exhaustive search of state space
- Limited by high computational complexity

Interactive verification

- Analysis reduces to proving a theorem in a logic
- Uses interactive theorem prover
- Requires more expertise

Limitations Formal Verification

- Appropriate only for control-intensive applications with interesting interaction among components
- Decidability and complexity remain obstacles; great progress in finding heuristics; flexibility in setting up the problem
- Falsification rather than verification: model, and not system, is verified; only stated requirements are checked
- Finding appropriate abstractions requires expertise

The Formal Methods Jungle

ACL2, ACP, ACSR, Action Semantics, Argos, ASM, ADLT, BDDs, B, Boyer-Moore, Caesar/Aldebaran, CCS, Circa, COLD, Coq, COSPAN, CSP, FDR2, CWB, DisCo, DC, Estelle, EVES, GIL, HOL, HyTech, IMPS, I/O Automata, ITL, Isabelle, JAPE, KIV, Kronos, LAMBDA, Larch, LeanTaP, LEGO, LOTOS, Lustre, MALPAS, Meije, Mizar, μ CRL, Murphi, NP-tools, Nqthm, Nuprl, OBJ, Otter, Petri Nets, Pi-calculus, Pobl, PRISM, ProofPower, PVS, RAISE, Rapide, Refinement Calculus, SAL, SDL, SGM, Signal, SMV, SPARK, SPIN, STeP, TAM, TAM97, Temporal-Rover, TLA, TPS, TRIO, TTM/RTTL, Unity, UPPAAL, VeriSoft, VDM, VIS, Z, ..

We will emphasize use of model checking, using UPPAAL, SAL and PRISM.

Course objectives

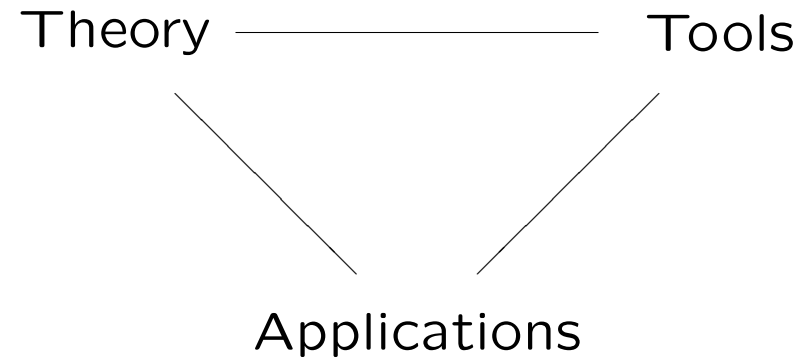
After successful completion participants are able to:

- recognize situations in which applications of model checking for specification and analysis may be useful,
- explain modelling frameworks and basic theory of finite state, real-time, and probabilistic automata,
- model (critical parts of) embedded systems as networks of automata,
- formalize desired properties in terms of automata or temporal logic, and
- use state-of-the-art tools for analysis of realistic problems.

In This Course We Will Use Three Model Checkers

- UPPAAL - timed automata
- SAL - symbolic and bounded model checking
- PRISM - probabilistic automata

The Trinity of Formal Methods



We will start from problems related to applications, try to use model checking tools to solve these, and study some theory to obtain the necessary background.

Timed Automata

- Model of finite automata enriched with real-values clock variables proposed by Rajeev Alur and David Dill in 1990



Timed Automata (cnt)

- Model checking tools under development since then; enormous progress has been made!
- Especially UPPAAL has become quite mature
- Dozens of industrial applications: embedded controllers, distributed algorithms and protocols, scheduling problems,...

Selected Applications: Embedded Systems

1. Architecture evaluation for a distributed in-car radio navigation system (Siemens VDO)
2. Analysis of data path design in copier (Océ)
3. Scheduling a lacquer production plant (Axxom)
4. MAC protocol for wireless sensor networks (Chess)
5. Throughput optimization for a wafer scanner (ASML)
6. Analysis of a car periphery supervision system (Bosch)
7. Scheduling of a steel plant (Sidmar)

Selected Applications: Communication Protocols

1. SHIM6, an internet protocol supporting host-based multihoming
2. IPv4 Zeroconf protocol for self configuration of IP network interfaces
3. IEEE 1394 tree identify
4. Audio control protocol of Philips
5. Biphase mark protocol